

Development of a High Performance Code for Hydrodynamic Calculations Using Graphics Processor Units

^{1,2}Andrey Sentyabov, ^{1,2}Andrey Gavrilov, ³Maxim Krivov, ^{1,2}Alexander Dekterev,
⁴Mikhail Pritula

¹Institute of Thermophysics SB RAS, Novosibirsk, Russia,

²Siberian Federal University, Krasnoyarsk, Russia,
sentyabov_a_v@mail.ru, gavand@yandex.ru, dekterev@mail.ru

³Lomonosov Moscow State University, Moscow, Russia,
m_krivov@cs.msu.su

⁴CTP PCP RAS, Moscow, Russia
pritmick@yandex.ru

Abstract. The paper presents the results of the implementation of computational algorithms of hydro-dynamics for using of graphics processor units. The implementation was carried out on the basis of in-house CFD code SigmaFlow. Numerical simulations were based on the solution of the Navier-Stokes equations using SIMPLE-like procedure. Discretization of the differential equations was based on the control volume method on unstructured mesh. In case of multiple CPU/GPU, parallel calculations were performed by means of domain decomposition. In GPU-version of the code, basic computational functions were implemented as CUDA kernels to perform on GPUs. The code has been verified using several test cases. Computational efficiency of several GPUs was compared to each other and to the modern CPUs. Modern GPU can increase the performance of calculations of CFD problems in more than 2 times compared to modern 6-core CPU.

Keywords: GPGPU, CUDA, MPI, CFD, numerical simulation, control volume method, SIMPLE, incompressible flow, domain decomposition

1 Introduction

Modelling of natural phenomena and industrial processes requires continuous growth of the computing performance. In recent, the performance of graphics processor units (GPU) has increased so much that they have become attractive for the scientific and engineering simulations. As a result, techniques of the GPGPU (General-Purpose computation on GPUs) have been developing. The high computational efficiency of graphical processor units leads to the wide application of GPUs in supercomputing systems [1]. There are many fields of the GPGPU calculations such as linear algebra [2],

molecular dynamics [3], aeromechanics [4, 5] and so on. The GPU performance is rapidly increasing and is much higher than the computational performance of central processor units (CPU).

Most algorithms of computational fluid dynamics for incompressible flows are based on the elliptic equation for the pressure correction. The algorithms include following main steps: a discretization of the pressure correction equation and velocity equations, solving the linear systems, corrections of the pressure and velocity fields. The implementation on the GPU only certain algorithm functions is not effective. All the data for the calculation are required to be stored in the GPU memory since their transfer between the GPU and the CPU takes a very long time. Consequently, all the main computational operations should be implemented on the GPU. The limited memory of the graphics processor units imposes serious restrictions on the computational problem in the case of single GPU. Multiple GPU systems allow computing the complex problem with fine mesh. In this case, a computational domain decomposition can be used for the parallel calculation on the multiple GPU. As known, an efficiency of the parallel calculations goes down as the number of the mesh cells increasing which is a result of the intensification of the data exchange between the computational units. In the case of graphics processor units, the speed of the data exchange is limited by the PCI-E bus.

The paper considers the realization of the GPU-version of an in-house computational fluid dynamic (CFD) code. The CFD code allows modeling the 3D steady and unsteady viscous incompressible flow in a complex computational domain. The main objectives of this paper are a demonstration of the problems, which can be solved by means of GPU, and comparison of the GPU and CPU performance for the incompressible flow modelling.

2 Mathematical Model

The implementation of GPGPU calculations is based on the in-house CFD code SigmaFlow [6]. Three-dimensional incompressible flow is described by the Navier–Stokes equations:

$$\nabla \cdot \mathbf{v} = 0 \quad (1)$$

$$\frac{\partial(\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = -\nabla p + \nabla \cdot \mathbf{T} \quad (2)$$

where viscous stress tensor is:

$$\mathbf{T}_{ij} = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \quad (3)$$

Model LES WALE (Large Eddy Simulation Wall Adapting Local Eddy-viscosity) is used for the simulation of turbulent flows. This version of the LES model is suitable for the simulation of the turbulent flow near the wall. The equations for the filtered velocity differ from the Navier–Stokes equations by additional subgrid stress tensor [7]:

$$\mathbf{T}_{ij}^t = \mu_{\text{sgs}} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \quad (4)$$

where μ_{sgs} is subgrid viscosity. In the LES WALE model, the subgrid viscosity is a function of the flow and is defined as follows [8]:

$$\mu_{\text{sgs}} = (C_W \Delta) \frac{\left(s_{ij}^d s_{ij}^d \right)^{3/2}}{\left(s_{ij} s_{ij} \right)^{5/2} + \left(s_{ij}^d s_{ij}^d \right)^{5/4}}, \quad s_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \quad (5)$$

$$s_{ij}^d = \frac{1}{2} \left(g_{ij}^2 + g_{ji}^2 \right) - \frac{1}{3} g_{kk}^2 \delta_{ij}, \quad g_{ij}^2 = g_{ik} g_{kj}, \quad g_{ij} = \frac{\partial v_i}{\partial x_j}$$

where Δ is a cell size, $C_W = 0.325$ is a model constant.

The numerical method is based on the finite volume method on an unstructured mesh. The distributions of a field and its gradient are used for the discretization of a differential equation. Gradient value in the center of the finite volume is determined by means of least-squares method [9].

Coupling of the pressure and velocity fields is one of the main challenge in the numerical modelling of the incompressible flows. A SIMPLE-like procedure on the collocated grids is used. There are many references concerning SIMPLE approach [10 – 12]. Collocated grids are used the same finite volumes for all variables (both pressure and velocity). This is the most efficient approach. To eliminate the pressure field oscillations the Rhie–Chow method [13] is applied.

Quick scheme [14] and Umist TVD [15] scheme are used for approximation of the convective term of the velocity equations. A second order scheme is used for the viscous term. Unsteady calculations are based on the implicit three-level second-order scheme. Linear systems for velocity equations are solved by means of incomplete factorization method DILU [16]. A variant of Krylov subspace iterative methods is used for solving the linear system of the pressure correction equation.

3 Software Implementation

Computational domain decomposition is used for parallel computations on the multiple CPU or GPU. Decomposition is splitting of the computational domain on subdomains. The each subdomain is handled by a separate computational process. The connection between the subdomains is realized by MPI interface. The distribution of the computational load on these processes should be uniform to obtain the highest performance. Partitioning of the computational nodes between the subdomains is performed by means of software MeTiS [17].

In the GPU-version of the code, all the computational operations are performed on GPU: calculation of the pressure and velocity gradients, discretization of the velocity equation and pressure correction equation, solving the linear systems, correction of the pressure and velocity according to SIMPLE procedure and calculation of the turbulent viscosity in case of turbulent flow modelling. Development of the GPU-versions of all the parts of the package was performed to prevent an excessive data transfer over the PCI-E bus. In this code, the data transfer from the GPU memory to the CPU main memory performs only in the MPI data exchange or data writing. Algorithms of the CUDA-kernels are identical to the corresponding CPU-method except for DILU method. Due to intensive use of atomic operations, CUDA Compute Capabilities 2.0 architecture is used for the implementation of these functions (CUDA-kernels). In order to achieve enough parallelism, the CUDA-kernels perform an operation on values at the each mesh cells or faces. Kernels take an array representing the field distribution as an argument and perform many identical operations on the elements of the arrays. Thus, the thread corresponds to the index of the mesh cell or the mesh face.

Similar to the CPU-version of the code, computational domain decomposition and MPI were used for the calculations by mean of the multiple GPU. In this case, each subdomain was calculated on the separated GPU.

4 Laminar Test Cases

The GPU-version of the code was verified with several test cases. Two laminar test cases were considered in this paper in detail. These are laminar flow in a cylinder with rotated endwall and unsteady laminar flow around a circular cylinder.

The swirled flow in the cylindrical container is produced by the endwall rotating with angular velocity Ω (Fig. 1). The endwall rotates the fluid by friction force. Centrifugal force throws the fluid near the rotated endwall to the periphery. Near the contrary endwall the flow returns to the center. Thus, a concentrated vortex is formed on the axis of the container. The flow is determined by the Reynolds number $Re = \Omega R^2/\nu$ and the ratio H/R , where ν is the kinematic viscosity, H is the height of the cylinder, R is the radius of the container. Two regime were considered. In the first one, the Reynolds number was $Re = 1800$ and the geometrical ratio was $H/R = 1$. In this case, the computational mesh included 800 thousand hexahedral cells. In the second regime, the Reynolds number was $Re = 2752$ and the geometrical ratio was $H/R = 3.25$. In this case,

a fine computational mesh was considered. The mesh included 10 million cells, which was concentrated near the wall.

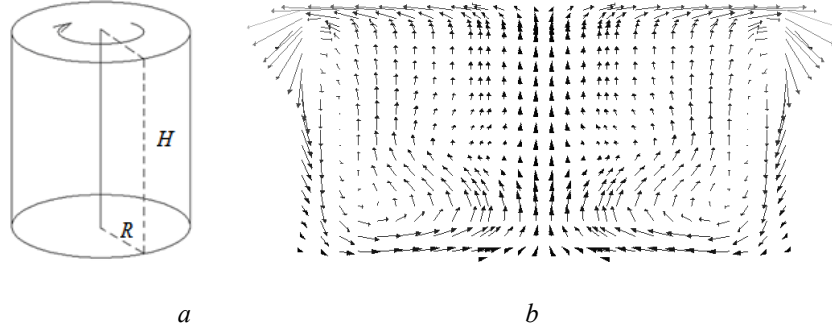


Fig. 1. The flow produced in a cylindrical container by a rotating endwall: a) the scheme; b) the flow in the central cross-section

For the regime $H/R = 1.0$ radial and tangential velocity components along the lines $r = 0.6R$ and $r = 0.9R$ were compared with the experimental data [18]. As Fig. 2a shows, the CPU and GPU code give the same results that are close to experimental data. In the regime $H/R = 3.25$, $Re = 2752$ experiment shows three-bubble vortex breakdown [19]. Vortex along the axis of the container directed from the rigid to the rotated endwall. The vortex core transforms (vortex breakdown). As a results, a recirculation zone (a “bubble”) forms on the axis. Then two more bubbles form near the center of the container. This regime corresponds to the very narrow range of the parameters. However, numerical results agree with experiment well.

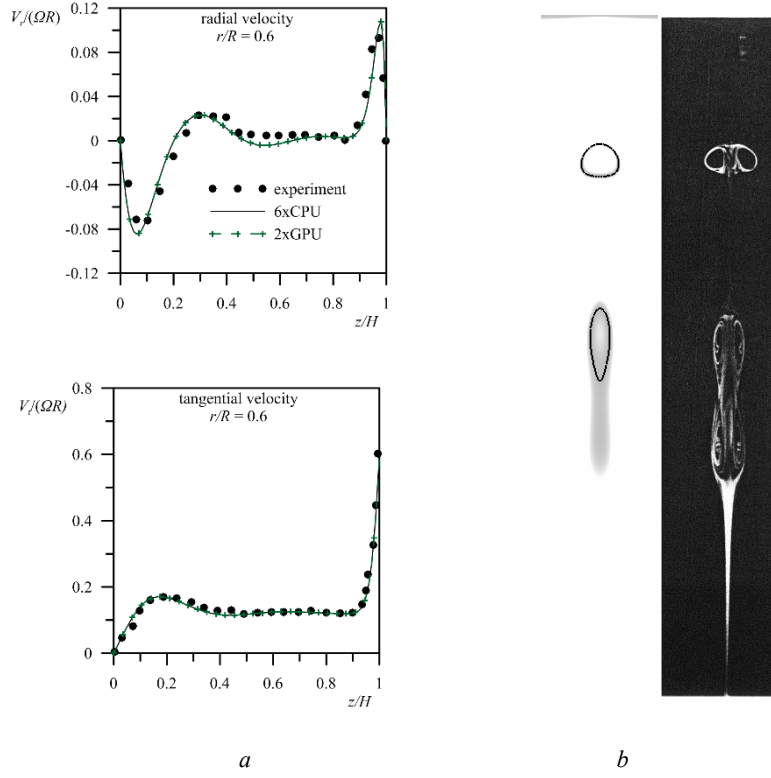


Fig. 2. The flow produced in the cylindrical container by a rotating endwall: *a*) $H/R = 1$, $Re = 1800$ (velocity components along the vertical lines, experiment [18]); *b*) $H/R = 3.25$, $Re = 2752$ (left: iso-line of zero-value of vertical velocity and stagnant zone; right: experimental photo [19]; rotated endwall locates on the bottom)

The time of the calculation was considered to compare computational efficiency of the CPU and GPU. The same number of iteration of the SIMPLEC procedure was fixed in the calculations (2000 iterations in case of $H/R = 1$, $Re = 1800$, and 20 000 iterations in case of $H/R = 3.25$, $Re = 2752$). The Fig. 3 shows that GPU Titan Black is two times faster than 6-cores CPU Core i7-5820k. Time of the calculation on the GPU Titan Black corresponds the system including two CPU. Fine mesh, including 10 million cells, required two GPU Titan Black due to memory limits.

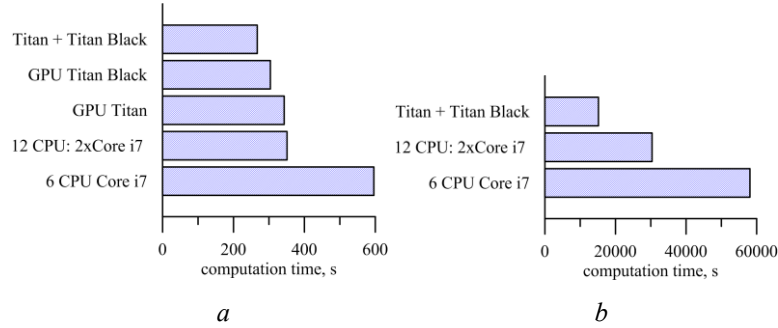


Fig. 3. Time of the calculation of the flow in the cylindrical container: *a*) $H/R = 1$, $Re = 1800$ (800 thousand cells); *b*) $H/R = 3.25$, $Re = 2752$ (10 million cells)

Second test case is unsteady flow around a circular cylinder (Fig. 4). The flow depends on the Reynolds number $Re = UD/\nu$, where U is the bulk velocity, D is the diameter of the cylinder, ν is the kinematic viscosity. The Reynolds number is 100. In this case, Karman vortex street is formed behind the cylinder (Fig. 5). Fig. 4a shows a scheme of the computational domain. The external boundary has size $D_{ext}=40D$. The length of the cylinder is $4D$. Uniform velocity distribution was set as inlet boundary conditions. Non-reflective boundary conditions was used on the outlet. Symmetry was used on the side walls. A number of the O-type meshes with concentration to the wall and to the wake included from 0.75 to 1.5 million cells. Time step is $\tau = 0.04T_{ref}$ where $T_{ref} = D/U_{in}$ is a reference time of the flow. Preliminary calculations showed Karman vortex street (Fig. 5a). Strouhal number (dimensionless frequency) of the vortex shedding is $St = fD/(U_{in})$ where f is the frequency of the vortex shedding, Hz. Numerical Strouhal number (see Table 1) agrees with experimental [20, 21] and another numerical results [22]. Averaged length of the recirculation zone also agrees with the results [22] well.

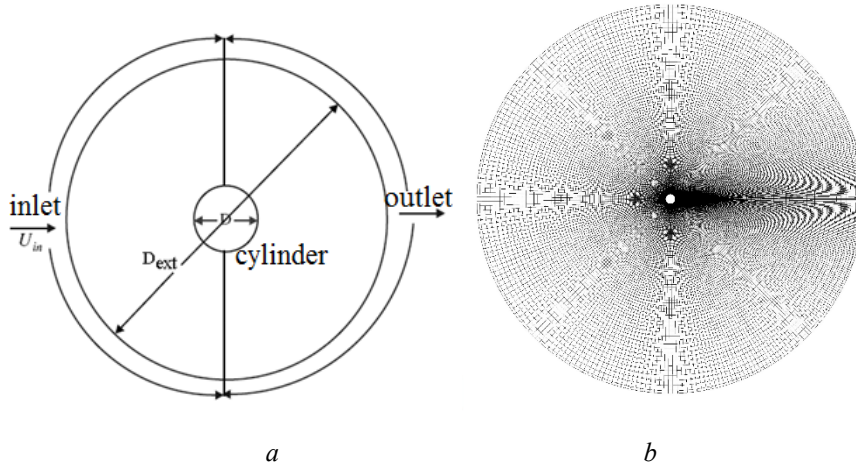


Fig. 4. Flow around a circular cylinder: *a*) the scheme; *b*) the computational mesh

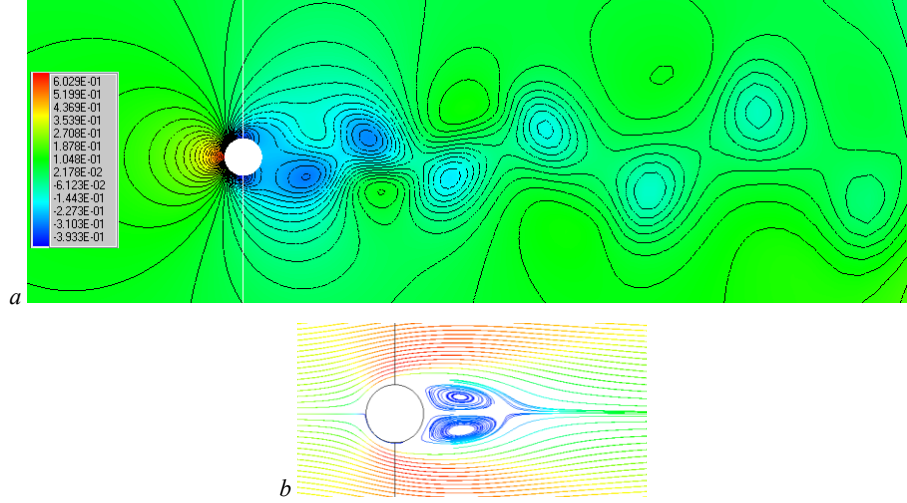


Fig. 5. Unsteady flow around a cylinder: *a)* Karman vortex street, visualized by instantaneous pressure distribution; *b)* streamlines of the averaged flow

Table 1. Flow around a cylinder: mean parameters

	St	L/D
calculations SigmaFlow-GPU	0.164	1.41
calculations Shoeybi (2010)	0.168	1.41
experiment Zdravkovich (1997)	0.165	—

For comparison of the computational efficiency, unsteady calculations of the flow around the cylinder were performed using uniform initial velocity field. Time of calculation of first $0.6T_{\text{ref}}$ (15 time steps) were used to compare computational efficiency of the GPUs. Number of iteration of SIMPLEC procedure per time step was fixed equal 30. Fig. 6. shows the computational efficiency of the different GPUs. As shown, the performance of the modern GPU is 2 – 3 times higher than the 6-cores CPU. The highest performance shows Titan Black, GeForce 1070 and GeForce 780Ti. GeForce series did not show much lower performance than Tesla series. On the other hand, Tesla GPGPUs have much more memory (Table 2). Memory is one of the major parameter for computational fluid dynamic because industrial and scientific problems requires very fine computational meshes. Parallel computations on modern GPUs allow meet these requirements.

Usually, single precision is used in computational fluid dynamics. However, in some cases single precision is insufficient. In the same time, GPU performance in double precision is lower than in single precision. Therefore, for this test case performance in

single and double precision was compared for different GPUs (Fig. 7.). Computational mesh included 0.75 million cell due to memory limits of the GPUs. Both Tesla and GeForce GPUs show computational time in double precision that is two times higher than in single precision. Thus, performance in double precision is determined primarily by memory bandwidth instead of calculations blocks.

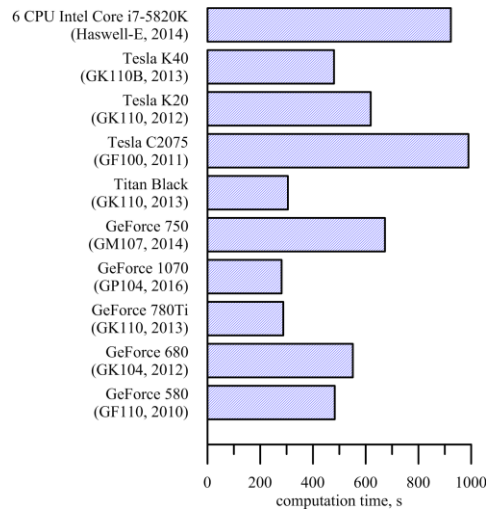


Fig. 6. Flow around a cylinder: the computation time (the mesh includes 1.5 million cells, single precision)

Table 2. Memory of the considered graphics processor units

GPU	Memory, Gb
Tesla K40	12
Tesla K20	5
Tesla C2075	3
GeForce 750	2
GeForce 1070	8
GeForce 780Ti	3
GeForce 680	2
GeForce 580	1.5
Titan Black	6

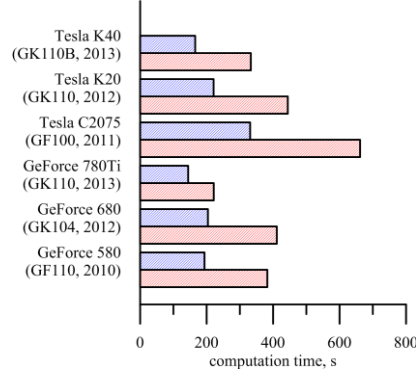


Fig. 7. Flow around a cylinder: the computation time (the mesh includes 0.75 million cells, single and double precision)

5 Turbulent Test Cases

In addition to laminar test cases, two turbulent problems were solved by means of LES WALE method. The first turbulent test case was the flow in a cubic cavity with a leading cover wall, which moves in the x -direction (Fig. 8a). The Reynolds number based on the cover wall velocity and the cavity length was $Re = 10^4$. The computational mesh included 1 million cells $100 \times 100 \times 100$ concentrated to the walls (Fig. 8b). Fig. 9 shows comparison of the averaged velocity along the central vertical line with the experimental data [23]. As shown, both CPU- and GPU-version of the code closely agrees with the experimental results. Comparison of the computational time shows that performance of two GPU Titan Black is 4 times higher than performance of two 6-core CPU Intel Core i7-5820k (Fig. 10).

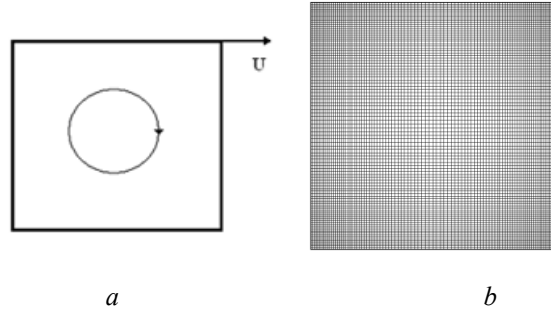


Fig. 8. Flow in the cavity: *a*) the scheme; *b*) the computational mesh

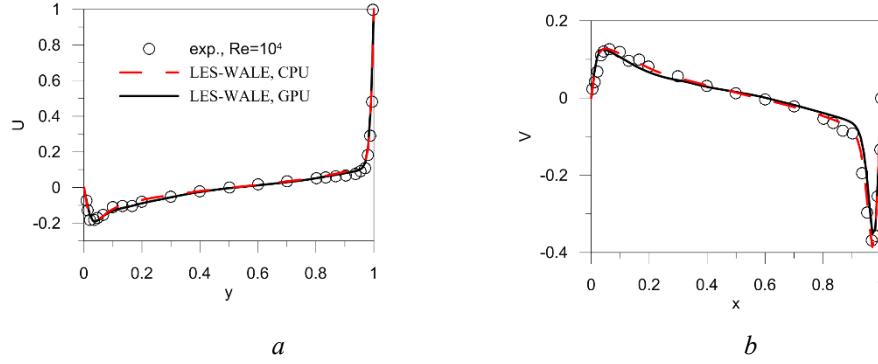


Fig. 9. Averaged velocity components along the central vertical line: *a*) horizontal velocity component; *b*) vertical velocity component

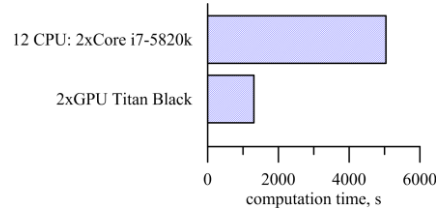


Fig. 10. The cavity: the computation time

Second test case was turbulent swirled flow in draft tube of a model hydraulic turbine. The calculations were based on the experimental data of the workshop Francis-99 [24] and numerical simulation [25]. Computational domain included draft tube with conical inlet (Fig. 11a). Averaged velocity profile on the inlet, obtained in the numerical investigations [25], was used as inlet boundary conditions. Part load operation mode was considered. The mesh included 7.8 million hexahedral cells (Fig. 11b). Dimensionless wall distance of the near wall nodes was $y_+ \approx 2$. Umist TVD scheme was used for the approximation of the convective terms. It is not sufficient for the appropriate LES calculations however it is reasonably for the estimation of the computational efficiency. Fig. 11c shows complex vortex structures in the draft tube. Comparison of the averaged velocity profiles behind the runner shows good agreement with experimental data (Fig. 12). As Fig. 13 shows, performance of two GPU Titan Black is 6 times higher than performance of 6-core CPU Intel Core i7-5820k. Performance of 4 GPU is 40% more than that of 2 GPU.

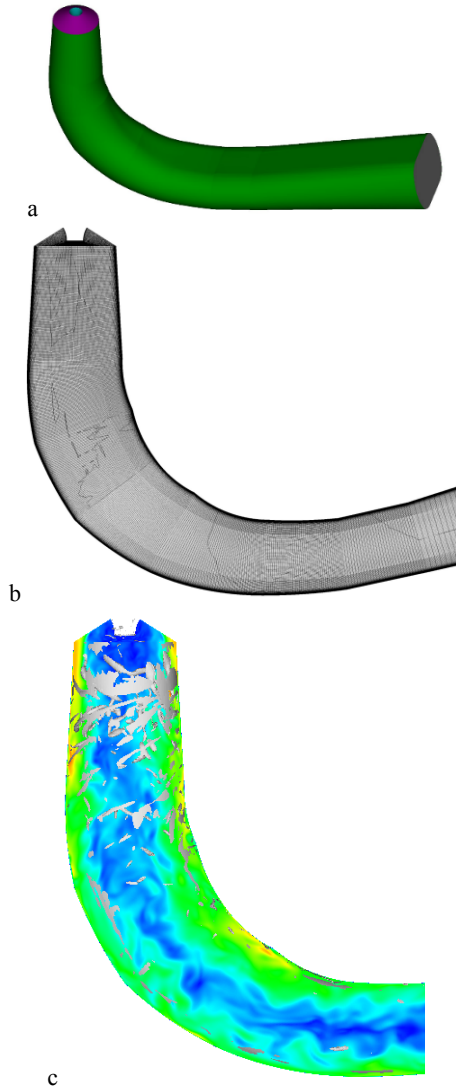


Fig. 11. Flow in the draft tube of the Francis-99 hydraulic turbine: a) the computational domain; b) the computational mesh in the central longitudinal cross-section; c) vortices visualized by the iso-surface of the Q -criterion (second invariant of the velocity gradient) and instantaneous velocity magnitude in the central cross-section

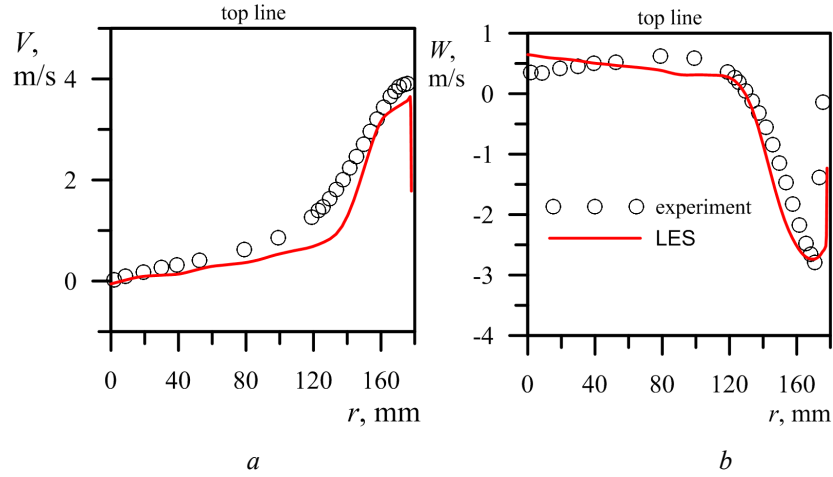


Fig. 12. Averaged velocity components in the Francis-99 draft tube: *a*) axial velocity; *b*) tangential velocity

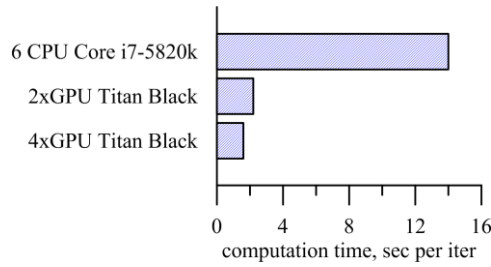


Fig. 13. Flow in the Francis-99 draft tube: the computation time per iteration of the SIMPLE procedure

6 Conclusions

Thus, a GPU-version of the CFD code was developed for parallel GPGPU simulation of the incompressible flows. All test cases calculations showed agreement with experimental and other numerical results. Such problems as steady and unsteady laminar flows, turbulent flow in the cavity and swirled turbulent flow in the part of the hydraulic turbine were considered. Most of the arrays were stored in the GPU memory and main operations of the SIMPLE procedure were performed on the GPU to provide the highest performance of the code.

The calculations showed high performance of the graphics processor units. Efficiency of the modern GPUs is 2 – 3 times higher than 6-core CPU. Parallel calculation with the multiple GPU systems allow overcoming the memory limits of a single GPU. In case of coarse mesh, parallel calculations on the multiple GPU is not efficient due to

the data exchange with a GPU. Memory is one of the major parameter for computational fluid dynamic because industrial and scientific problems requires very fine computational meshes. Therefore, a code for parallel computations on modern GPUs can be a useful tool of CFD calculations.

Acknowledgments

The work was financially supported by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Region Science and Technology Support Fund, research project No16-41-243033.

References

1. <https://www.top500.org/lists/2016/06/>
2. Li R. and Saad Y. GPU-accelerated preconditioned iterative linear solvers. *The Journal of Supercomputing*, 2013, 63(2):443 – 466.
3. Rinaldi P.R., Dari E.A., Venere M.J., and Clausse A. A Lattice-Boltzmann solver for 3D Fluid simulation on GPU. *Simulation Modelling Practice and Theory*, 2012, 25: pp. 163 – 171
4. Corrigan A., Camelli F.F., Lohner R. and Wallin J. Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 2011, 66(2): 221 – 229.
5. Waltz J. Performance of a three-dimensional unstructured mesh compressible flow solver on NVIDIA Fermi-class graphics processing unit hardware. *International Journal for Numerical Methods in Fluids*, 2013, 72(2): 259 – 268
6. Gavrilov A.A., Dekterev A.A., and Sentyabov A.V., Modeling of Swirling Flows with Coherent Structures Using the Unsteady Reynolds Stress Transport Model // *Fluid Dynamics*, 2015, Vol. 50, No 4, pp. 471 – 482.
7. Pope, S.B. *Turbulent flows* / S.B. Pope. — New York : Cambridge University Press, 2000. — 771 p.
8. Nicoud F., Ducros F. Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor // *Flow, Turbulence and Combustion*, 1999, Volume 62, Issue 3, pp 183–200. DOI: 10.1023/A:1009995426001
9. Mavriplis J. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes // *AIAA-Paper 2003-3986*, June 2003.
10. Ferziger J.H., Peric M. *Computational methods for fluid dynamics*. —Springer, 2002, 423 pp.
11. Moukalled F., Darwish M. A Unified Formulation of the Segregated Class of Algorithms for Fluid Flow at All Speeds // *Numerical Heat Transfer, Part B*. — 2000. — Vol. 37, N 2. — P. 227–246.
12. Patankar S. *Numerical heat transfer and fluid flow*. — New York. Hemisphere Publishing Corporation, 1980, 197 p.
13. Rhie C.M., Chow W.L. A Numerical Study of the Turbulent Flow Past an Isolated Airfoil with trailing Edge Separation // *AIAA Journal* — 1983. — Vol. 21. — P. 1525–1532.
14. Leonard B.P. A stable and accurate convective modeling procedure based on quadratic upstream interpolation // *Comp. Math. Appl. Mech. Eng.*, 1979, Vol. 19., pp. 59 – 98.

15. Leschziner, M.A. Upstream monotonic interpolation for scalar transport with application to complex turbulent flows / Leschziner M.A., Lien F.S. // *Int. J. Num. Meth. Fluids*. — 1994. — Vol. 19, N 6. — P. 527–548.
16. Barrett R., Berry M., Chan T. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. — s.l.: SIAM, 1994. — P. 141.
17. Karypis G., Kumar V. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs // *SIAM Journal on Scientific Computing*, 1999, Vol. 20, No. 1, pp. 359–392.
18. J.A. Michelsen. *Modeling Incompressible Rotating Fluid Flow*. – Technical University of Denmark, 1986, Ph.D. thesis.
19. M.P. Escudier. Observation of the flow produced in a cylindrical container by a rotating endwall // *Exp. in Fluids*, 1984, Vol. 2, No 4, p. 189–196.
20. Zdravkovich M.M. *Flow around circular cylinders. Vol 1: Fundamentals*, Oxford University Press (1997)
21. Rajani B., Kandasamy A. and Majumdar S. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling* 33, 1228-1247 (2009).
22. Shoeybi M., Svard M., Ham F.E. and Moin P. An Adaptive Implicit-Explicit Scheme for the DNS and LES of Compressible Flows on Unstructured Grids. *Journal of Computational Physics* 229, 5944-5965 (2010).
23. Akula B., Roy P., Razi P., Anderson S. and Girimaji S. Partially-Averaged Navier-Stokes (PANS) Simulations of Lid-Driven Cavity Flow—Part 1: Comparison with URANS and LES // *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 2015, Vol. 130, pp.421 – 430.
24. <http://www.ltu.se/research/subjects/Stromningslara/Konferenser/Francis-99>.
25. Minakov A.V. Sentyabov A. V., Platonov, D.V., Dekterev A.A. and Gavrilov A.A. Numerical modeling of flow in the Francis-99 turbine with Reynolds stress model and detached eddy simulation method. *Journal of Physics: Conference Series*, 579(1), 2015.